Write a C program to find FIRST( ) - predictive parser for the given grammar

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAX 10

char grammar[MAX][MAX];  // Stores the grammar rules
char first[MAX][MAX];    // Stores FIRST sets
int ruleCount;           // Number of rules

// Function to check if a character is a terminal
int isTerminal(char c) {
   return !isupper(c);
}

// Function to add a symbol to FIRST set (if not already present)
void addToFirst(char *firstSet, char symbol) {
   int len = strlen(firstSet);
   for (int i = 0; i < len; i++) {
      if (firstSet[i] == symbol)
         return;  // Avoid duplicates
   }
   firstSet[len] = symbol;
   firstSet[len + 1] = '\0';
}

// Function to compute FIRST set for a given non-terminal
void computeFirst(char nonTerminal, char *firstSet) {
   for (int i = 0; i < ruleCount; i++) {
      if (grammar[i][0] == nonTerminal) {  // Find rules for this non-terminal
```

```
            char *rhs = &grammar[i][3];  // Right-hand side of production
            if (isTerminal(rhs[0]) || rhs[0] == 'ε') {
                addToFirst(firstSet, rhs[0]);  // Add terminal or epsilon
            } else {
                // If RHS starts with non-terminal, compute FIRST recursively
                char subFirst[MAX] = "";
                computeFirst(rhs[0], subFirst);
                for (int j = 0; subFirst[j] != '\0'; j++) {
                    addToFirst(firstSet, subFirst[j]);
                }
            }
        }
    }
}

// Main function
int main() {
    printf("Enter the number of grammar rules: ");
    scanf("%d", &ruleCount);
    getchar(); // Consume newline

    printf("Enter the grammar rules (in format A->B|c, use 'ε' for epsilon):\n");
    for (int i = 0; i < ruleCount; i++) {
        fgets(grammar[i], MAX, stdin);
        grammar[i][strcspn(grammar[i], "\n")] = '\0';  // Remove newline
    }

    printf("\nFIRST Sets:\n");
    for (int i = 0; i < ruleCount; i++) {
        char nonTerminal = grammar[i][0];
```

```
        // Check if FIRST(nonTerminal) is already computed
        int alreadyComputed = 0;
        for (int j = 0; j < i; j++) {
            if (grammar[j][0] == nonTerminal) {
                alreadyComputed = 1;
                break;
            }
        }
        if (alreadyComputed) continue;

        char firstSet[MAX] = "";
        computeFirst(nonTerminal, firstSet);

        printf("FIRST(%c) = { ", nonTerminal);
        for (int j = 0; firstSet[j] != '\0'; j++) {
            printf("%c ", firstSet[j]);
        }
        printf("}\n");
    }

    return 0;
}
```

Input:

Enter the number of grammar rules: 2

Enter the grammar rules:

E->TR

T->+TR|ε

Output:

```
PS C:\Users\valli>  & 'c:\Users\valli\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-leoaiq3u.wxv' '--stdout=Microsoft-MIEngine-Out-rbgwdnaq.kbx' '--stderr=Microsoft-MIEngine-Error-4vwijydi.vei' '--pid=Microsoft-MIEngine-Pid-dlwriteq.qvz' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the number of grammar rules: 3
Enter the grammar rules (in format A->B|c, use 'ε' for epsilon):
E->TR
T->+TR|ε

FIRST Sets:
FIRST(E) = { + }
FIRST(T) = { + }
FIRST() = { }
```